

PJT명	프로젝트 기획과 협업	
단계	[Django 개발]	
진행일자	2025.11.07	
예상 구현 시간	필수기능	5H
	추가기능	2H
	심화기능	1H

## 1. 목표

- 데이터를 생성, 조회, 수정, 삭제할 수 있는 Web application을 제작한다.
- Django Model과 ORM에 대하여 이해한다.
- Django Form에 대하여 이해한다.
- Many to one relationship(N:1)에 대하여 이해한다.
- 프로젝트를 기획하고, 기획 내용을 바탕으로 구체적인 요구사항을 작성할 수 있다.
- 기획서를 통해 개발 단계에서 참조할 수 있는 문서를 작성할 수 있다.

## 2. 준비사항

### 1) 제공 파일

- 팀별 기획서 양식 word 파일

### 2) 개발언어 및 툴

- Python 3.11
- HTML, CSS
- Visual Studio Code
- Google Chrome

### 3) 필수 라이브러리 / 오픈소스

- Django 5.2

- Bootstrap 5.3

### 3. 작업 순서

- 1) 팀원과 같이 요구사항을 확인하고, GitLab에 프로젝트를 생성한다.
  - 프로젝트 이름은 06-pjt로 지정한다.
  - 각 반 담당 강사님을 Maintainer로 설정한다.
- 2) 팀원과 합의하여 협업 방식을 결정하고, 요구사항을 구현한다.
- 3) 작성한 코드들을 정리하고, README를 작성한다.
  - .gitignore 파일을 활용하여 불필요한 파일 및 폴더는 제출하지 않는다.
- 4) README 작성이 완료되면 심화 학습을 진행한다.
- 5) 제출 기한에 맞춰 모든 산출물이 GitLab에 업로드 될 수 있도록 한다.

## 4. 요구사항

AI 기반 도서 분석과 창작 지원 통합 솔루션 서비스를 구축하려고 한다. 다양한 장르와 형태의 도서 데이터를 수집 및 관리하고, 이를 기반으로 AI 기반 도서 분석 기능을 설계 및 구현한다. 또한 작가들에게 아이디어 발상 지원, 플롯 구성 제안, 문장 교정 및 스타일 개선 등 창작 전반에 걸친 다양한 AI 기반 창작 지원 기능을 제공한다. 또한 도서에 대한 사용자 리뷰 및 감상평 공유 커뮤니티 기능을 제공하여, 사용자들이 활발하게 소통하고 정보를 교환할 수 있는 기능을 제공한다. 사용자는 자신이 읽은 도서를 평가하고, 다른 사용자의 리뷰를 참고하여 다음 도서를 선택하는데 도움을 받을 수 있다. 나아가, 관심 도서 목록을 맞춤형으로 구성하는 등 다양한 편의 기능을 제공한다. 팀원과 상의하여 아래 요구사항을 만족할 수 있도록 요구 사항 명세서를 작성 및 구현해보자.

도서를 읽은 후 감상을 공유할 수 있는 기능을 만드는 단계이다. 도서 데이터의 생성, 조회, 수정, 삭제가 가능하며, 해당 도서를 읽은 후 감상을 게시글의 형태로 생성, 조회, 수정, 삭제가 가능한 애플리케이션을 완성해보자.

- 요구사항 예시(참고용)
  - 아래의 내용을 참고하여 추가적인 아이디어에 대해 요구사항을 추가 또는 수정하여 기능을 구현한다. 단, **필수 기능은 구현해야 하며, 수정할 수 없다.**

번호	분류	요구사항명	요구사항 상세	우선순위
기능적 요구사항				
F01	프로젝트	프로젝트 구성	도서 커뮤니티 서비스 구현을 위한 Django 프로젝트 및 앱 생성	필수
F02	books	Model Class	도서 데이터를 데이터베이스에 저장할 수 있도록 Django Model 클래스 구현	필수
F03	books	Model Class	도서에 대한 게시글 데이터를 데이터베이스에 저장할 수 있도록 Django Model 클래스 구현	필수
F04	books	Form Class	사용자가 입력한 데이터 검증을 위한 Django Form 클래스 구현	필수
F05	books view 함수	index	전체 도서 데이터를 조회하는 view 함수 구현	필수
F06	books view 함수	create	도서 데이터를 입력할 수 있는 UI를 제공하고, 데이터에 따라 도서 데이터를 저장하는 view 함수 구현	필수
F07	books view 함수	detail	단일 도서 데이터를 조회하는 view 함수 구현	필수
F08	books view 함수	update	도서 데이터를 입력할 수 있는 UI를 제공하고, 선택한 도서 데이터를 수정하는 view 함수 구현	필수
F09	books view 함수	delete	단일 도서 데이터를 삭제하는 view 함수 구현	필수
F10	books view 함수	thread_create	게시글 데이터를 입력할 수 있는 UI를 제공하고, 데이터에 따라 게시글 데이터를 저장하는 view 함수 구현	필수
F11	books view 함수	thread_detail	게시글 데이터를 조회하는 view 함수 구현	필수
F12	books view 함수	thread_update	게시글 데이터를 입력할 수 있는 UI를 제공하고, 데이터에 따라 게시글 데이터를 수정하는 view 함수 구현	필수
F13	books view 함수	thread_delete	게시글 데이터를 삭제하는 view 함수 구현	필수
F14	AI 활용	커버 이미지	쓰레드 작성 시 커버 이미지를 생성형 AI로 생성하기	도전
F15	AI 활용	정보 생성 및 저장	생성형 AI API를 활용하여 작가 정보	도전

			검색 및 생성	
비기능적 요구사항				
NF01	프로젝트 관리	Git 활용	개발자 간 작업 충돌이 일어나지 않게끔 Git을 활용하여 프로젝트 관리	필수
NF02	유지보수	URL 구성	URL이 변경될 때 유지보수가 용이하도록 구현	필수
NF03	보안	HTTP Method 허용	허용된 HTTP Method를 사용하는 요청만 허락하도록 구현	필수

## 1) 프로젝트 관리 (필수)

Git의 기능을 적극적으로 활용하여, 여러 개발자가 동시에 개발할 수 있는 환경을 조성한다.

- 요구사항 번호: NF01
- 팀원과 합의하여 branch 생성 원칙 정리하기
  - Gitflow, GitHub Flow 등을 참고하여 원칙을 정할 수 있음
  - branch 운영 중 문제상황이 발생할 경우 원인에 대한 분석을 꼭 진행할 것
- 팀원과 합의하여 commit 내역 기록 원칙 정리하기
  - 기능 개발, 버그 수정 등 그 목적이 명확히 드러날 수 있도록 commit 메시지를 작성할 것
  - 어느 시점에 commit을 남길지에 대하여 충분한 합의를 이룰 것
- 프로젝트 종료 시 초기에 세운 원칙이 잘 지켜졌는지 점검
  - README에 초기에 세운 원칙을 설명하여 정리할 것
  - 생성했던 branch 들에 대한 용도를 정리할 것
  - 작성된 commit들을 GitLab에서 확인하여 스크린샷으로 README에 포함시킬 것

## 2) 프로젝트 및 앱 (필수)

도서 데이터를 생성, 조회, 수정, 삭제할 수 있는 Django 프로젝트를 만든다. 명시된 요구사항 이외 서비스를 위해 필요하다고 생각되는 기능 등은 자유롭게 구현해도 무관하며, Bootstrap을 활용하여 자유롭게 스타일링 한다.

- 요구사항 번호: F01, NF02, NF03
- 프로젝트 이름: mypjt
- 앱 이름: books
- 페이지간 이동 편의성을 갖추기 위한 Navbar 구성
  - books의 index, create의 기능들에 접근하기 편하도록 링크를 제공
  - 다양한 페이지에서 도달할 수 있도록 여러 페이지에 동일하게 출력되도록 구현

## A. books

도서 데이터를 관리하기 위한 앱이다.

- 요구사항 번호: F02, F03, F04
- Model 클래스
  - Book: 도서 데이터를 나타내는 Model. 도서 제목, 설명, 회원 리뷰 평점, 저자를 저장할 필드 4개 지정
  - Thread: 도서에 대한 게시글을 나타내는 Model. 게시글 제목, 본문, 독서일, 생성 시각, 수정 시각을 저장할 필드 5개 지정
  - Book과 Thread는 1:N 관계를 맺고 있어야 함.
- Form 클래스
  - 각 Model 클래스에 대하여 데이터 검증, 저장, 에러메시지 등을 관리하기 위해 적절한 ModelForm 사용
- 각 세부 기능들을 위한 view 함수를 구성
  - index, create, detail, update, delete, thread\_create, thread\_detail, thread\_update, thread\_delete
  - URL이 변경될 가능성을 고려하여 유지보수에 유리한 형태로 구성
  - 각 view 함수가 허용하는 HTTP Method에만 호출되도록 구현



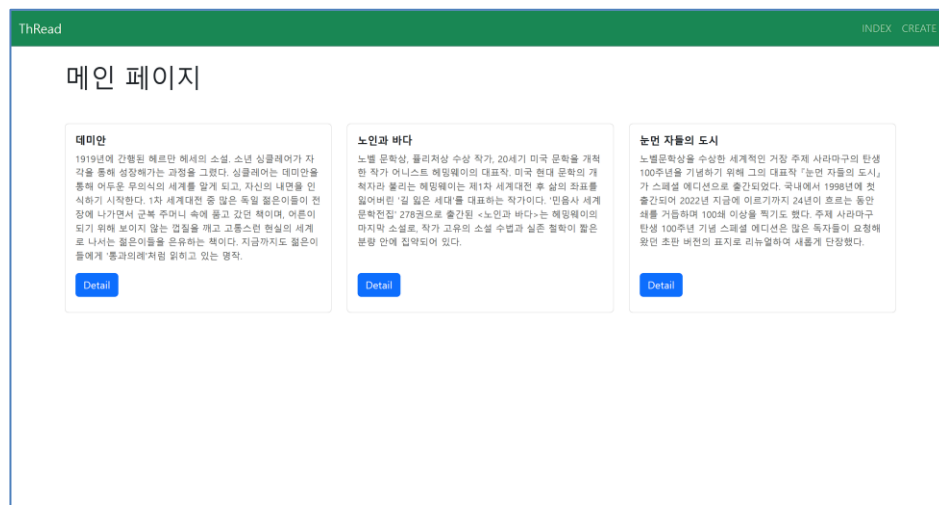
### 3) books 앱 view 함수

도서 데이터를 생성, 조회, 수정, 삭제하는 역할을 하는 books 앱을 구현하자.

#### A. index

전체 도서 데이터를 조회하기 위한 index view 함수를 구현한다.

- 요구사항 번호: F05
- 전체 도서 데이터가 조회되어야 함
- 특정 도서를 선택하여 detail 페이지로 이동할 수 있는 링크가 제공되어야 함
  - 링크의 문구와 UI는 자유롭게 구성
- 출력 예시



## B. create

도서 데이터를 입력하기 위한 UI를 제공하고, 입력된 데이터를 저장하는 create view 함수를 구현한다.

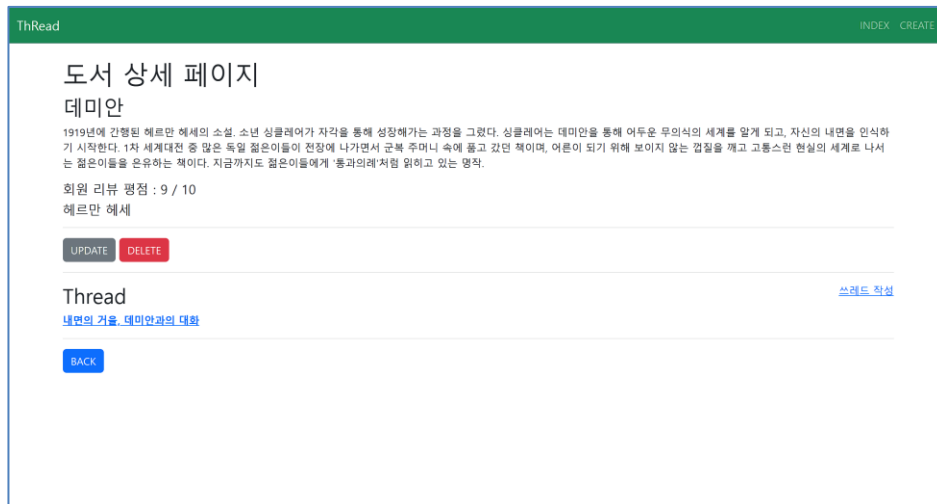
- 요구사항 번호: F06
- Book 모델의 데이터를 입력할 수 있는 UI 구현
  - 유효하지 않은 데이터가 저장되지 않도록 구현
  - 유효한 데이터의 경우 저장 후 작성된 도서의 detail 페이지로 redirect
  - UI 디자인은 자유롭게 구성
- 같은 view 함수에서 HTML 제공과 데이터 생성이 같이 이뤄지도록 구현
  - GET 요청을 받을 경우: HTML 제공
  - POST 요청을 받을 경우: 형식에 맞는 데이터는 저장, 형식에 맞지 않는 데이터는 오류를 페이지에 표시
- 출력 예시



## C. detail

단일 도서 데이터를 조회하는 detail view 함수를 구현한다.

- 요구사항 번호: F07
- index 페이지에서 도서를 선택했을 때 이동됨
  - 선택한 해당 도서의 정보를 바탕으로 내용이 출력되어야 함
- 도서 제목, 설명, 회원 리뷰 평점, 저자 데이터를 출력
  - 도서를 수정하기 위한 페이지로 이동하는 링크 구현
  - 도서를 삭제하는 버튼 구현
  - 이전 페이지(index)로 이동하는 링크 구현
  - UI 디자인은 자유롭게 구성
- 게시글 목록 및 작성 UI 출력
  - 게시글 제목이 목록으로 보이도록 구현
  - 게시글을 작성할 수 있는 페이지로 이동하는 링크 구현
- 출력 예시



## D. update

도서 데이터를 수정하기 위한 UI를 제공하고, 입력된 데이터를 바탕으로 대상 도서 데이터를 수정하는 update view 함수를 구현한다.

- 요구사항 번호: F08
- detail 페이지에서 수정하기 링크를 통해 이동됨
  - detail 페이지에서 제공되었던 도서 정보를 바탕으로 기능이 동작되도록 구현
- Book 모델의 데이터를 입력할 수 있는 UI 구현
  - 이전 도서의 데이터가 미리 입력되어 있도록 구현
  - 유효하지 않은 데이터가 저장되지 않도록 구현
  - 유효한 데이터의 경우 저장 후 작성된 도서의 detail 페이지로 redirect
  - UI 디자인은 자유롭게 구성
- 같은 view 함수에서 HTML 제공과 데이터 수정이 같이 이뤄지도록 구현
  - GET 요청을 받을 경우: HTML 제공
  - POST 요청을 받을 경우: 형식에 맞는 데이터는 저장, 형식에 맞지 않는 데이터는 오류를 페이지에 표시
- 출력 예시

ThRead INDEX CREATE

### 도서 수정 페이지

Title: 데미안

Description: 1919년에 간행된 헤르만 헤세의 소설. 소년 싱클레어가 자각을 통해 성장해가는 과정을 그린다. 싱클레어는 데미안을 통해 어두운 무의식의 세계를 알게 되고, 자신의 내면을 인식하기 시작한다. 1차 세계 대전 중 많은 독일 젊은이들이 전장에 나가면서 군복 주머니 속에 품고 있던 책이며, 어른이 되기 위해 보이지 않는 감정을 깨고 고통스런 현실의 세계로 나서는 젊은이들을 은유하는 책이다. 지금까지도

Customer review rank: 9

Author: 헤르만 헤세

저장

## E. delete

도서 데이터를 삭제하는 기능을 제공하는 delete view 함수를 구현한다.

- 요구사항 번호: F09
- detail 페이지에서 삭제하기 버튼을 통해 호출됨
  - detail 페이지에서 제공되었던 도서 정보를 바탕으로 기능이 동작되도록 구현
  - 도서를 작성한 사용자만 삭제할 수 있도록 구현
  - POST 요청에 대해서만 작동하도록 구현
  - 삭제가 된 이후에는 메인 페이지로 redirect

## F. thread\_create

게시글 데이터를 입력하기 위한 UI를 제공하고, 입력된 데이터를 저장하는 thread\_create view 함수를 구현한다.

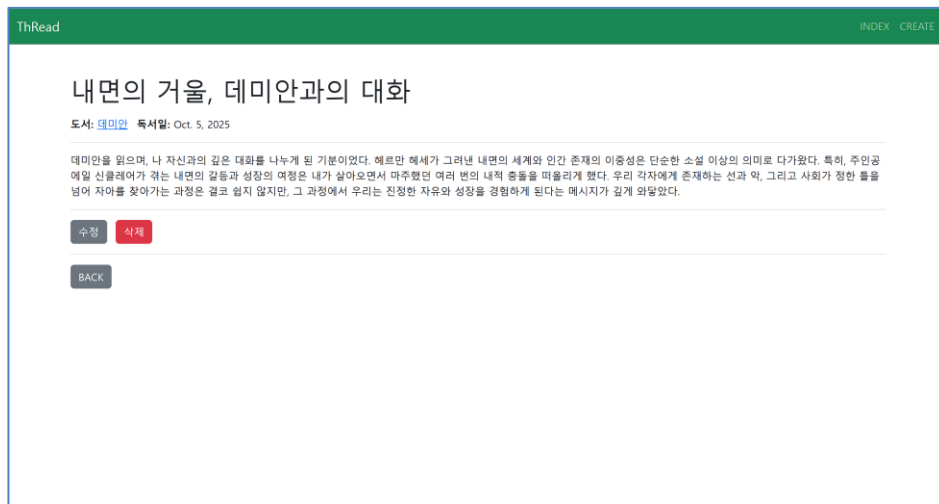
- 요구사항 번호: F10
- Thread 모델의 데이터를 입력할 수 있는 UI 구현
  - 유효하지 않은 데이터가 저장되지 않도록 구현
  - 유효한 데이터의 경우 저장 후 작성된 게시글 detail 페이지로 redirect
  - UI 디자인은 자유롭게 구성
- 같은 view 함수에서 HTML 제공과 데이터 생성이 같이 이뤄지도록 구현
  - GET 요청을 받을 경우: HTML 제공
  - POST 요청을 받을 경우: 형식에 맞는 데이터는 저장, 형식에 맞지 않은 데이터는 오류를 페이지에 표시
- 출력 예시

The screenshot displays a web browser window with a green header bar. On the left of the header is the text 'ThRead', and on the right are the links 'INDEX' and 'CREATE'. The main content area is titled '쓰레드 작성' (Thread Creation). Below the title, there is a form with the following elements: a '제목' (Title) input field, a large '내용' (Content) text area, and a '등록일' (Registration Date) field with a calendar icon. At the bottom of the form, there are two buttons: a green '작성' (Write) button and a red '취소' (Cancel) button.

## G. thread\_detail

단일 게시물 데이터를 조회하는 thread\_detail view 함수를 구현한다.

- 요구사항 번호: F11
- detail 페이지에서 게시글을 선택했을 때 이동됨
  - 선택한 게시글의 정보를 바탕으로 내용이 출력되어야 함
- 게시물 제목, 본문, 독서일 데이터를 출력
  - 게시글을 수정하기 위한 페이지로 이동하는 링크 구현
  - 게시글을 삭제하는 버튼 구현
  - 이전 페이지(detail)로 이동하는 링크 구현
  - UI 디자인은 자유롭게 구성
- 출력 예시



## H. thread\_update

게시글 데이터를 수정하기 위한 UI를 제공하고, 입력된 데이터를 바탕으로 대상 게시글 데이터를 수정하는 thread\_update view 함수를 구현한다.

- 요구사항 번호: F12
- thread\_detail 페이지에서 수정하기 링크를 통해 이동됨
  - thread\_detail 페이지에서 제공되었던 게시글 정보를 바탕으로 기능이 동작되도록 구현
- Thread 모델의 데이터를 입력할 수 있는 UI 구현
  - 이전 게시글의 데이터가 미리 입력되어 있도록 구현
  - 유효하지 않은 데이터가 저장되지 않도록 구현
  - 유효한 데이터의 경우 저장 후 수정된 게시글의 thread\_detail 페이지로 redirect
  - UI 디자인은 자유롭게 구성
- 같은 view 함수에서 HTML 제공과 데이터 수정이 같이 이뤄지도록 구현
  - GET 요청을 받을 경우: HTML 제공
  - POST 요청을 받을 경우: 형식에 맞는 데이터는 저장, 형식에 맞지 않는 데이터는 오류를 페이지에 표시



## ● 출력 예시

ThReadINDEX CREATE

쓰레드 수정

제목 내면의 거울, 데미안과의

데미안을 읽으며, 나 자신과의 깊은 대화  
를 나누게 된 기분이었다. 헤르만 헤세가  
그려낸 내면의 세계와 인간 존재의 이중  
성은 단순한 소설 이상의 의미로 다가왔  
다. 특히, 주인공 예달 신클레어가 겪는 내  
면의 갈등과 성장의 여정은 내가 살아가  
면서 마주했던 여러 번의 내적 충동을 따  
올리게 했다. 우리 각자에게 존재하는 선  
과 악, 그리고 사회가 창힌 틀을 넘어 자  
아를 찾아가는 과정은 결코 쉽지 않지만,

내용

독서일 2025-10-05

전송 취소

## I. thread\_delete

게시글 데이터를 삭제하는 기능을 제공하는 thread\_delete view 함수를 구현한다.

- 요구사항 번호: F13
- thread\_detail 페이지에서 삭제하기 버튼을 통해 호출됨
  - thread\_detail 페이지에서 제공되었던 게시글 정보를 바탕으로 기능이 동작되도록 구현
  - POST 요청에 대해서만 작동하도록 구현
  - 삭제가 된 이후에는 관계를 맺고 있는 도서 데이터의 detail 페이지로 redirect

#### 4) 도전 과제

생성형 AI 도구를 활용하여 도전과제 요구사항을 해결해보자.

- 코드 생성, 아이디어 구상, 문제 해결 방법 탐색 등 다양한 방식으로 활용할 수 있다.
- 사용할 생성형 AI 서비스는 자유롭게 선택한다.
  - 제공되는 GPT API Key를 활용할 경우, 기능별로 정해진 모델을 활용한다.
- 최종 결과물은 AI 생성 내용을 바탕으로 직접 수정 및 개선하여 적용해본다.

보조 수단으로 활용하되, 능동적인 자세로 학습에 임할 것.

- 최종적인 이해와 적용은 자기 주도적 학습을 통해 이루어지며, 배운 내용을 스스로 기록하고 정리하며 학습 효과를 높일 것.

## A. 생성형 AI를 활용한 게시글 커버 이미지 생성

쓰레드 작성 시 커버 이미지를 생성형 AI로 생성해보자.

- 요구사항 번호: F14
- 도서 제목, 저자 이름, 독서 게시글의 감정을 분석해 키워드 5개를 추출
  - 추출을 위해 생성형 AI 활용
  - 제공되는 GPT API Key를 활용할 경우, 반드시 gpt-5-nano 모델을 사용
- 추출한 키워드를 기반으로 이미지 생성을 위한 프롬프트 작성
  - 작성된 프롬프트를 통해 어떤 이미지를 그려야 할지 결정 후 생성형 AI에게 전달
  - 제공되는 GPT API Key를 활용할 경우, 반드시 dall-e-3 모델을 사용
- 생성된 이미지를 파일 시스템에 저장 후 Thread 모델에 필드를 추가하여 저장
  - cover\_image 필드를 추가하여 저장
- 출력 예시



## B. 생성형 AI API를 활용한 작가 정보 생성 및 저장

생성형 AI API를 활용하여 작가 정보를 생성하고 저장해보자.

- 요구사항 번호: F15
- 도서를 등록할 때 BookForm으로 입력받은 작가 이름을 위키피디아 API의 쿼리로 전달해 작가 정보, 프로필 사진 검색
- 사용자 입력 데이터와 위키피디아 검색 정보를 바탕으로, GPT API를 활용하기
  - 제공되는 GPT API Key를 활용할 경우, 반드시 gpt-5-nano 모델을 사용
  - 작가 정보와 대표 작품 목록 요청
  - JSON 형태로 응답하도록 프롬프트 작성
- Book 모델에 필드를 추가하여 저장
  - author\_info와 author\_works 필드를 추가하여 저장
- 추가한 필드를 도서 detail 페이지에 출력
  - 작가 정보 영역 아래에 "작가 대표작과 작가 정보는 AI가 생성한 정보로, 정확하지 않을 수 있습니다."라는 안내 문구 추가
- 출력 예시



## 5. 프로젝트 기획

이번 단계에서는 관통 프로젝트(PJT)의 전체 설계 방향을 청사진 수준으로 기획하는 것을 목표로 한다.

지금까지 학습한 기술 스택(Django, DB, HTML/CSS, API, AI 활용 등)을 종합하여 팀별로 아이디어를 구체화하고, 실제 서비스로 발전 가능한 형태로 정리한다.

향후 진행할 관통 프로젝트의 데모를 확인하고, 이를 기반으로 서비스 아이디어를 도출한 뒤, **팀별 기획서(docx)를 작성하여 제출**해야 한다.

본 단계의 핵심은 단순한 코드 구현이 아니라,

- "어떤 문제를 해결하는 서비스를 만들 것인가?"
- "우리 팀의 서비스가 어떤 AI 기능과 결합될 수 있을까?"

를 명확히 정의하는 것이다.

제공되는 팀별 기획서 양식을 요약하면 다음과 같다

단계	수행 내용	비고
1	관통 PJT 데모를 확인하고, 전체 서비스 흐름을 이해한다.	데모 사이트 참고
2	팀별로 어떤 서비스를 개발할지 기획한다. 기존 관통 PJT 주제를 응용하거나 확장하여, 우리 팀만의 차별화된 기능을 제안한다.	신규 기능 제안 가능
3	서비스에 적용할 AI 기능을 구상한다. 예를 들어, 추천 시스템, 자동 분석, 요약, 생성형 응답 등 다양한 방향을 검토한다.	OpenAI / Gemini / HuggingFace / 공공데이터 활용 가능
4	관련된 데이터셋을 탐색한다. (전자정부 데이터, 공공 API, 크롤링 등 활용 가능)	데이터 수집 방식 기획

5	유사 서비스나 레퍼런스를 조사하여 벤치마킹한다.	URL, 주요 기능 요약 포함
6	모든 내용을 '팀별 기획서.docx' 파일에 정리하고 GitLab 에 업로드한다.	파일명: 팀번호_기획서.docx

## 6. 참고자료

- Django 5.2  
<https://www.djangoproject.com/start/overview/>
- Python OpenAI Library  
<https://platform.openai.com/docs/libraries?language=python>  
<https://github.com/openai/openai-python>
- OpenAI 이미지 생성 API 문서  
<https://platform.openai.com/docs/guides/image-generation>
- Excalidraw  
손으로 그린 듯한 스타일의 웹 기반 협업 화이트보드  
<https://excalidraw.com/>
- diagrams.net (구 draw.io)  
흐름도, ERD, UML 등 다양한 다이어그램을 온라인에서 무료로 작성할 수 있는 그래픽 편집 도구  
<https://app.diagrams.net/?src=about>
- Figma  
UI/UX 설계, 프로토타이핑, 팀 간 실시간 피드백을 지원하는 웹 기반 협업 디자인 툴  
<https://www.figma.com/ko-kr/>
- Notion  
노트, 문서, 위키, 프로젝트 관리 등을 하나의 워크스페이스에서 작성할 수 있도록 통합된 생산성 툴  
<https://www.notion.com/ko>



## 7. 결과

제출 기한은 진행일 18시까지이므로 제출 기한을 지킬 수 있도록 한다. 제출은 GitLab을 통해서 이뤄진다.

- 산출물과 제출

- 단계별로 구현 과정 중 학습한 내용, 어려웠던 부분, 새로 배운 것들 및 느낀 점을 상세히 기록한 README.md
- 완성된 각 문제 별 소스코드 및 실행 화면 캡처본
- 프로젝트 이름은 06-pjt로 지정 및 각자 제출
  - 한 명의 GitLab 계정에 Git 저장소 Push 및 작업
  - 나머지 인원은 제출 시 해당 저장소를 Fork 하여 제출
- 각 반 담당 강사님을 Maintainer로 설정

- 끝 -